COMPUTING SCIENCE

Formal Modelling of Railway Safety and Capacity

Alexei Iliasov and Alexander Romanovsky

TECHNICAL REPORT SERIES

No. CS-TR-1444 January 2015

TECHNICAL REPORT SERIES

No. CS-TR-1444 January, 2015

Formal Modelling of Railway Safety and Capacity

A. Iliasov and A. Romanovsky

Abstract

Development of future railway systems requires a rigorous modelling of safety and capacity conducted in an integrated way. Supported by EPSRC and Rail Safety and Standards Board the SafeCap project laid the foundations for overcoming challenges to railway capacity without undermining rail network safety. The main outcome of the project is the SafeCap Toolset, which relies on a formal Domain Specific Language, safety verification and capacity simulation methodologies. The work was conducted in close cooperation with Siemens Rail Automation and evaluated using the layouts of a number of UK stations. The Toolset is being further actively developed and evaluated in a series of industrial and impact acceleration projects.

Printed and published by Newcastle University,
Computing Science, Claremont Tower, Claremont Road,
Newcastle upon Tyne, NE1 7RU, England.

Bibliographical details

ILIASOV, A., ROMANOVSKY, A.

Formal Modelling of Railway Safety and Capacity
[By] A. Iliasov, A. Romanovsky,

Newcastle upon Tyne: Newcastle University: Computing Science, 2015.

(Newcastle University, Computing Science, Technical Report Series, No. CS-TR-1444)

Added entries

NEWCASTLE UNIVERSITY
Computing Science. Technical Report Series. CS-TR-1444

Abstract

Development of future railway systems requires a rigorous modelling of safety and capacity conducted in an integrated way. Supported by EPSRC and Rail Safety and Standards Board the SafeCap project laid the foundations for overcoming challenges to railway capacity without undermining rail network safety. The main outcome of the project is the SafeCap Toolset, which relies on a formal Domain Specific Language, safety verification and capacity simulation methodologies. The work was conducted in close cooperation with Siemens Rail Automation and evaluated using the layouts of a number of UK stations. The Toolset is being further actively developed and evaluated in a series of industrial and impact acceleration projects.

About the authors

Alexander (Sascha) Romanovsky is a Professor in the Centre for Software Reliability. He is the leader of the Dependability Group at the School of Computing Science.

His main research interests are system dependability, fault tolerance, software architectures, exception handling, error recovery, system verification for safety, system structuring and verification of fault tolerance.

Prof Romanovsky is now the Principle Investigator of the TrAmS-2 EPSRC/UK platform grant on Trustworthy Ambient Systems (2012-16) and of the EPSRC/RSSB research project SafeCap on Overcoming the Railway Capacity Challenges without Undermining Rail Network Safety (2011-14), and the Co-investigator of the EPSRC PRiME program grant (2013-18) and of the FP7 COMPASS Integrated Project (2011-14).

Suggested keywords

VERIFICATION
TOOLS
RAILWAY SIGNALLING

Formal Modelling of Railway Safety and Capacity

Alexei Iliasov and Alexander Romanovsky

CSR, Newcastle University

Abstract *Development of future railway systems requires a rigorous modelling of safety and capacity conducted in an integrated way. Supported by EPSRC and* Rail Safety and Standards Board *the SafeCap project laid the foundations for overcoming challenges to railway capacity without undermining rail network safety. The main outcome of the project is the SafeCap Toolset, which relies on a formal Domain Specific Language, safety verification and capacity simulation methodologies. The work was conducted in close cooperation with Siemens Rail Automation and evaluated using the layouts of a number of UK stations. The Toolset is being further actively developed and evaluated in a series of industrial and impact acceleration projects.*

1 Introduction

The Rail Technical Strategy 2012 produced by the UK Technical Strategy Leadership Group (TSLG) sets out a number of challenging objectives to be achieved by the UK railway industry in the coming 40 years (TSLG 2012). As part of this work a series of the SafeCap projects have been funded to develop theories, methods and tools that address the challenges of improving railway capacity at the same time providing strong guarantees of system safety. The approach taken by the project is motivated by the needs to deal with the railway systems/networks of growing complexity, to reduce their development time, and to increase the confidence in the products developed.

One of the main decisions we made in this work was to use formal methods to model systems, stations, layouts and control tables, and to formally verify their consistency and safety. Even though the railway industry is the main success story in accepting formal methods, their application is still patchy. One of the barriers is the high cost of training and deployment. To address this issue we developed a graphical Domain Specific Language that has formal semantics and allows us to fully hide formal methods and tools. This is complemented by a high performance automated verification back-end capable of verifying large stations automatically.

The capacity is evaluated by simulation of the formal models and the results are shown in terms of the Domain Specific Language.

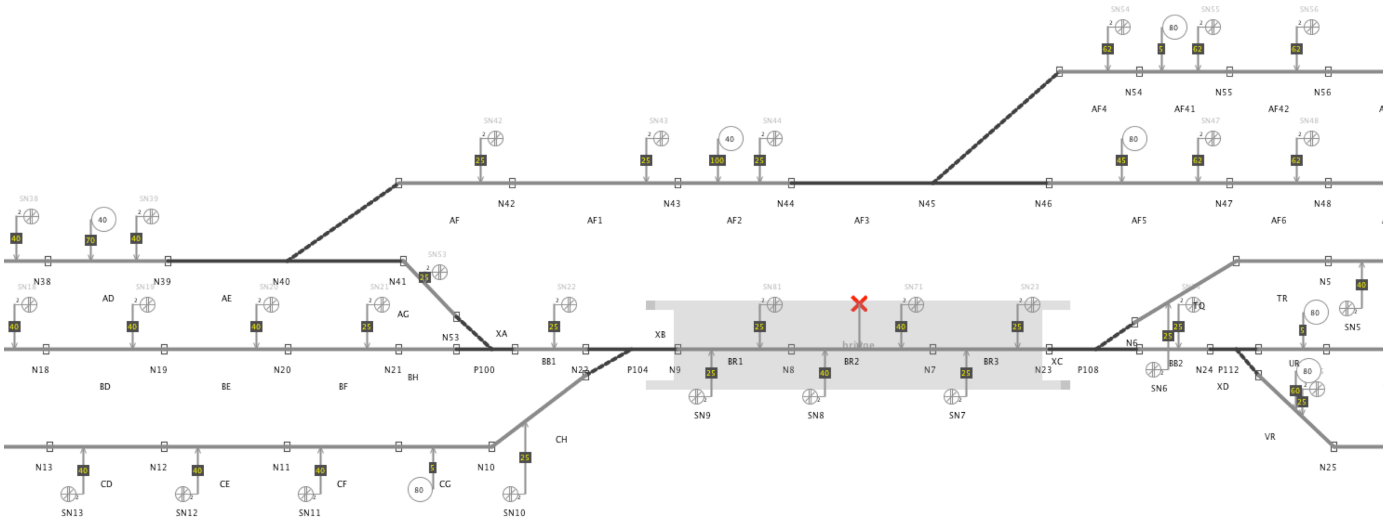
In this work we are relying on the extensive experience we gained in the FP7 DEPLOY Integrated Project on developing the Rodin toolset supporting the Event-B method and in deploying it in Bosch, SAP, Siemens Transportation Systems, and other companies (Romanovsky and Thomas 2013). This experience has helped us to develop an extensible SafeCap Eclipse-based environment supporting the work of railway signalling engineers in designing stations/junctions that are safe and have the improved capacity.

2 Problems and objectives

Let us discuss the principal problems and objectives of railway signalling verification. Within the hierarchy defined in (Fokkink and Hollingshead 1998) we focus exclusively on the middle interlocking layer leaving out details of the lower layer of physical equipment functioning and the upper layer of railway logics and exploitation largely out of the view. We identify five kinds of railway safety verification concerns. The first three are the fundamental safety properties that may be reasoned about at a specification level abstracting away from minute details of physical track topology and the setting in which the track is laid. The remaining two require consideration of concrete track geometry, topography, train exploitation characteristics and prospective service requirements.

A schema must be free from collisions. A collision happens when two trains occupy the same part of a track. Reasoning about collisions must take into an account concrete topology, requires an explicit train notion, the definition of laws of train movement and assumptions about train driver (either human or automatic) behaviour. Note that if train drivers choose to ignore whatever means of indication of track occupation states are available to them (e.g., track side signals) there is nothing preventing two trains from colliding. Hence, the absence of collisions is ensured by demonstrating the compatibility of specific topology, signalling and certain driving rules.

The basic safety mechanism is that of route locking and holding (see Figure 1 below). A train is given permission to enter an area of a railway once there is a continuous and safe path through this area assigned exclusively to this train. Such a path is normally called a *route* and is delineated by *signals* - either physical trackside signals with lamps or conceptual signals displayed to a driver via a computer screen. Two-aspect signals (red/green or stop/proceed) are positioned at the maximum braking distance from each other and this defines the smallest train separation. 3- and 4- and higher aspect signalling allows trains to come closer by advising drivers on the safe speed and the extent of free track available in front.



point P100 : clear(XA)
route SN39_SN42 : clear(AE, AF) ∧ reverse(P102)
route S23_S22 : clear(BH, XA, BB1) ∧ clear(AE, AG) ∧ reverse(P102)
route S22_S81 : clear(XB, BR1, BR2, BR3, XC) ∧ normal(P104)
... : ...

Fig.1. A part of mid-size junction topology and an excerpt from its control table. Route `S21_S22` consists of three train detection circuits: `BH`, `XA` and `BB1`. A train detection circuit is a part of a railway (a sub-graph in an abstract topology) with some equipment capable of reporting the presence or absence of a train in this part. To avoid derailment, the movement of point `P100` requires that circuit `XA` is clear. This, in its turn, requires that routes `S53_S22` and `S21_S22` are not set. To control speed on a curved track, the layout uses fixed speed limits (circles with numbers) and approach speed control. One example of the latter is a control table mandating that a train travelling over the route `S71_S23` occupies `BR3` for at least 15 seconds.

When a route is locked, all the movable equipment such as *points* or level crossings must be set and detected in a position that would let a train safely travel on its desired route. They must remain locked in such a state until the train passage is positively confirmed.

A schema must be free from derailments. A derailment may happen when a train moves over a point that is not set in any specific direction and thus may move under a train. To avoid this, a point must be positively confirmed to be *locked* before a train may travel over it. In a control table one writes a condition defining when a point reconfiguration may happen.

Another reason for derailment is driving a train through a curve at an unsafe speed. As a train goes over a curve, the combination of gravitational, centripetal and centrifugal forces exerts a rolling force on train carriages and a substantial lateral force on rails. This effect can be mitigated by track canting although no single canting is a perfect fit for all train types. Hence, enforcing a safe speed limit before a train enters a curved track area is an essential safety consideration. There are several ways of doing this. One is a static speed limit. This can be a signboard warning a driver or an electronic signal sent to an on-board computer. A speed restriction may be also enforced by signalling: a signal does not switch into a permissive aspect until a train is detected to occupy some preceding detection circuit for a duration time. A combination of such time duration and track length gives an upper train speed limit.

Physical layout properties. A range of properties pertinent to safety requires analysis of land topography over which track is build. As one example, it must be ensured that physical signals have certain minimum sighting distance giving a sufficient time for a driver to react. Sometimes tracks are so close together that carriages of a train going through a curve may come into a contact with carriages of a train located on a parallel track. A signalling engineer must identify and protect such areas (known as *fouling points*) via signalling rules. Further examples include gradients at stopping points (e.g., signals) that may be unsafe for heavy trains, parts of track susceptible to landslips, debris on the track due to nearby trees and overpasses, and so on. An important consideration is the spacing between signals and speed restriction signs: it must be possible for all trains to brake within the given limits to meet signal or speed limit restriction. Signal positioning and speed restriction would be wastefully conservative if one does not consider specific properties of traffic, in particular train acceleration and braking performance.

Quality of service. It is never sufficient to consider the safety aspect of a railway in isolation from its performance. Indeed, setting all signals permanently to red (stop) state trivially satisfies all the safety concerns discussed above. As a less extreme example, there could be a signalling mistake preventing or hindering train progress but not violating safety properties. Typically, when signalling a station or a junction, an engineer would have access to a provisional timetable. A timetable defines traffic class and station calling and dwelling times. It must be ensured that signalling is able to accommodate such traffic with some extra margin for unac-

counted or delayed traffic. Simulation of train runs is the common way to check quality of service requirements.

3 Safety verification

Figure 3 depicts the interaction between a signaling engineer and a verification tool. There is a conceptual barrier between a railway model that an engineer interacts with and the model handled by formal verification tools.

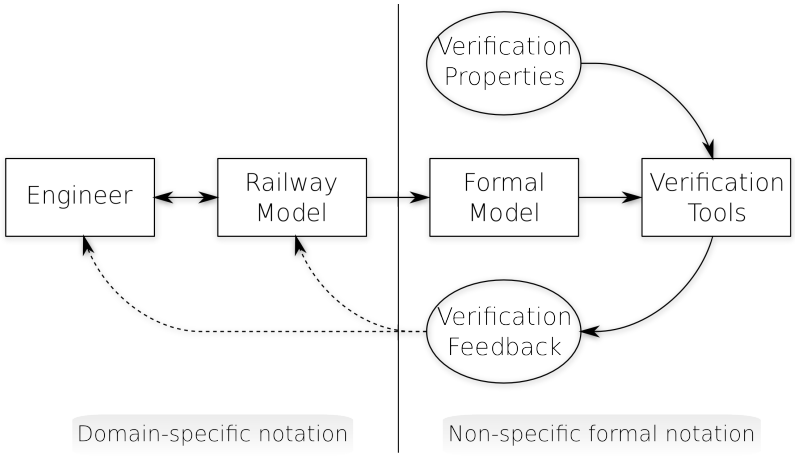


Fig. 3. Formal verification in railway domain: principal actors and flow

Let us now look into the main approaches to signalling verification approaches used in the industry and proposed by academia. In practice, several techniques are often combined to complement each other's strengths.

Manual review. Just as compilation of control tables is often a manual process, verification may also be accomplished via a carefully set up but otherwise manual review procedure. In most cases, to facilitate legibility, control tables are written in a highly structured tabular form following a common standard, i.e., UK Railway Group Standard GK/RT 0202 (RGSOnline 2014) although historic and regional peculiarities are not uncommon. One possible arrangement is having one company to design signalling and a competing company to verify it. The reasoning is that this way both parties are incentivized to do their best.

Manual review is a slow process with very high requirements to reviewers' expertise. It does not deliver any objective proof of safety. At the same time, it does not suffer from any limitations of a formal verification process.

Simulation. Railway industry widely employs railway simulation tools. These range from coarse-grained simulation of a national railway network to a detailed simulation of various aspects of mechanical performance of specific engines and

carriages in a combination with specific rail and ballast types. Verification concerns span from analysis of digital communication protocols connecting trains and regional control to stressing of tunnels and bridges by passing trains.

Simulation is widely applied for timetable optimisation and interactive 3D simulation is sometimes used for driver training.

RailSys (RailSys 2014) and OpenTrack (OpenTrack 2014) are two of the well-known simulation suites applied in timetable optimisation and general analysis of signalling performance.

The main attraction of simulation is that it does not require deep understanding of railway functioning. Simulation tools present many aspects of railway performance in an intuitive, visual manner helping to quickly obtain the big picture of overall layout and signalling performance. There is, however, no guarantee of safety as simulation can only ever consider a tiny proportion of all scenarios.

Model checking. The safety challenge of railways and the fact that collision and derailment properties may be dealt with within the setting of discrete, inertia-less train movement makes railway safety verification especially appealing for formal method practitioners. The principal idea of railway model checking is quite simple: a model of train movement laws is combined with the definitions of track topology and signalling rules. A model-checking tool attempts to go through all or many execution scenarios to confirm that unsafe scenarios are ruled out. The list of modelling notations used in this setting is practically endless. Notable examples include Coloured Petri nets (Janczura 1998), process algebra CSP (Winter 2002), a continuation work based on the model-based notation ASM (Winter and Robinson 2003), an algebraic language Maude (Hagalisletto et al. 2007) and the B Method together with ProB model checking tool (Leuschel and Butler 2003).

Almost all model-checking approaches allow automatic instantiation of template models making application of model checking relatively straightforward for engineers. Many tools are able to report a sequence of steps leading to a safety violation. While model checkers are able to analyse many more scenarios than a simulator this comes at a price of reduced expressiveness (i.e., inability to reason about track geometry) and proof certificate is generally not ultimate: there could be a false negative (i.e., the absence of an error report in case an error is present but not discovered) when a model is too large to analyse exhaustively.

Theorem proving. Model checking imposes limitations on the model size and performs best with a relatively limited logical language. Theorem proving overcomes these limitations and offers potentially unlimited opportunities for verifying safety with the utmost level of rigour. Theorem proving is not necessarily an all-manual process: there is a large and successful community developing automated theorem provers (TPTP 2014). At the moment, automated prover support is best in the domain of first order logic and set theory; an attempt at reasoning about continuous train dynamics is likely to require an intervention by a highly skilled verification expert - the kind of people mostly found in academia. From our experience, even reasoning about track geometry is surprisingly difficult as this is a problem outside of the typical application domain of verification tools. One success story with theorem proving is the on-going application of B method in the

railway domain (Essame and Dolle 2007). J.-R. Abrial has published methodological guidelines on an economical use of basic logic and set theory to reason about railway safety in a discrete setting (Abrial 2006).

Theorem proving, even with excellent tool support, requires a high level of expertise in formal verification and mathematical modelling. The semantic gap between logic and railway concepts is formidable. This leads to generally low productivity (but we should notice efforts like the BART tool for automatic refinement of B models (Burdy 1999), difficulties in interpreting tool feedback, and posing verification statements in a manner convincing to a non-expert reviewer.

4 Safety verification in SafeCap

The purpose of the SafeCap Toolset is to enable railway engineers to analyse complex junctions by experimenting with signalling rules, signalling principles, track topology, safety limits (e.g., speed limits for points and crossings) while receiving an on-line feedback from automated verification and analysis tools.

We have built the Toolset around Eclipse - a mature and extensible IDE framework. We used Eclipse Modelling framework (EMF) to realise our Domain Specific Language (Iliasov and Romanovsky 2012). One important consideration was the ability to benefit from the extensive EMF ecosystem which offers a toolkit for model manipulation and the construction of graphical and textual editing tools. Apart from the editing tools, the main components of the Toolset are transformation patterns, model-based animation, simulation and verification (see Figure 4).

We have applied the Event-B modelling notation and its refinement methodology to develop a theory of safe railway. This theory explicitly describes train movements, signal operation and point's control. It does not, however, deal with any specific topology or control table. The proof of safety (we consider absence of collisions and derailments, and protection of flanks[1]) is done for some class of topologies and control tables. The proof of the Event-B model, although challenging, is done once and for all.

An important by-product is the set of axiomatic conditions characterising the class of *safe topologies* and *safe control tables*. To establish that a given track topology and control table are safe we only need to check that they do not contradict the mentioned axiomatic conditions. We do not need to redo the proofs of Event-B model. Safety verification is accomplished by putting together the definition of a concrete topology, control table and the axiomatic conditions derived by the Event-B model. If a *constraint solver* does not find a contradiction in logical statements encoded by this composition then the concrete topology and control table are deemed safe. Returning to the Event-B domain, the absence of contradic-

[1] Protection of the movement of a train across a junction that prevent any other unauthorised movement coming into contact with it.

tion established by a constraint solver means that our generic Event-B model of train behaviour is refined by a model instantiated with the given track topology and control table.

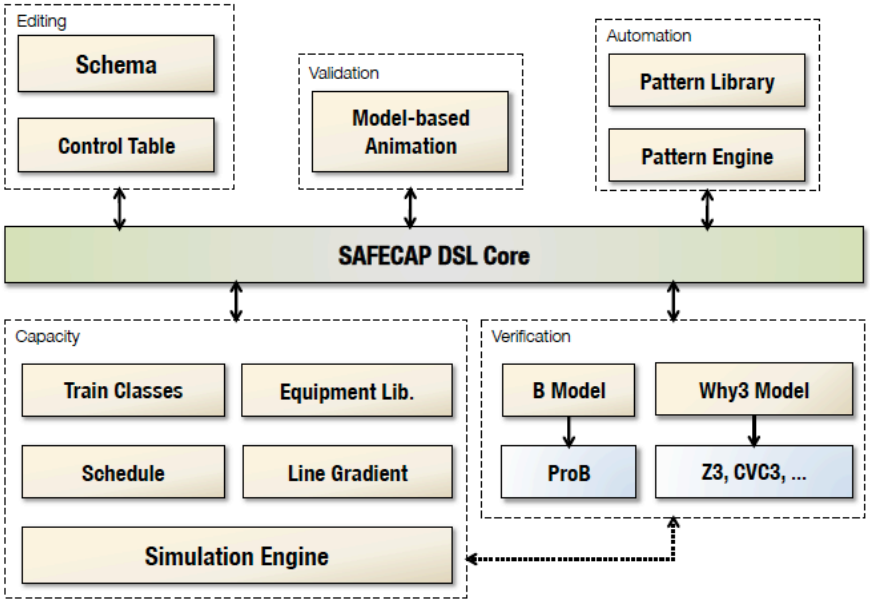


Fig.4. The architecture of the SafeCap Toolset

Schema topology and control table theories come in the form of a list of first order logic predicates; they do not define any state transitions or dynamic behaviour but rather well-formedness requirements to objects describing track topology and control table.

For constraint solving, we make use of two sets of formal notations and tools: B together with ProB (Leuschel and Butler 2003) and Why3 (Bobot et al. 2011). In the short term, we aim to benefit from their complementary strengths; in a longer term, the dual verification path provides a logical redundancy that makes a low-level encoding or tool bug unlikely to be left undiscovered.

4.1 Event-B

We apply the Event-B formal modelling notation (Abrial 2010) to specify and verify railway signalling. Event-B belongs to a family of state-based modelling languages that represent a design as a combination of state (a vector of variables) and state transformations (computations updating variables).

An Event-B development starts with the creation of a very abstract specification. A cornerstone of the Event-B method is the stepwise development that facilitates a gradual design of a system implementation through a number of correctness-preserving *refinement* steps. The general form of an Event-B model (or *machine*) is shown in Figure 5. Such a model encapsulates a local state (program variables) and provides operations on the state. The actions (called *events*) are characterized by a list of local variables (parameters) `vl`, a state predicate `g` called *event guard*, and a next-state relation `S` called *substitution* or event *action*.

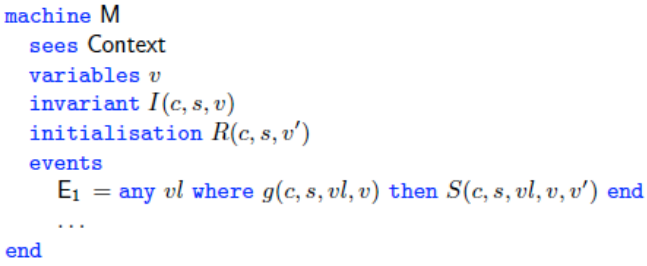


Fig.5 Event-B machine structure

Event parameters and guards may be omitted leading to syntactic short-cuts starting with keywords `when` and `begin`.

Event `g` defines the condition when an event is *enabled*. Relation `S` is given as a generalised substitution statement (Abrial 1996) and is either deterministic (`x := 2`) or non-deterministic update of model variables. The latter kind comes in two notations: selection of a value from a set, written as `x :∈ {2, 3}`; and a relational constraint on the next state `v'`, e.g., `x | x' ∈ {2, 3}`.

The `invariant` clause contains the properties of the system, expressed as state predicates, that must be preserved during system execution. These define the *safe states* of a system. In order for a model to be consistent, invariant preservation is formally demonstrated. Data types, constants and relevant axioms are defined in a separate component called *context*.

Model correctness is demonstrated by generating and discharging *proof obligations* - theorems in the first order logic. There are proof obligations for model consistency and for a refinement link - the forward simulation relation - between the pair of *abstract* and *concrete* models.

4.2 Discrete driving model

The discrete driving model is an Event-B model capturing train, signal and point behaviour. It proves that the described behaviour is contained within a certain

safety envelope by formulating and proving, through a number of refinement steps, *safety invariants* corresponding to the first three verification objectives of Section 2. This model gives a formal definition of principal phenomena observed in railway operation: train movement, route reservation, point locking, route cancellation and so on.

To construct the proof we have used Event-B (Abrial and Mussat 1998) and the Rodin Toolset (Rodin 2014). Train driving rules are encoded by Event-B events - atomic state transitions - so that the overall model defines a state transition system. The safety properties are stated as a system invariant: a subset of possible states where the dangerous situations may not occur. The proof is done inductively by examining the effect of each event on a given safety property and discharging relevant proof obligation (first-order logic theorems).

The model in Figure 6 illustrates the notation and modelling style of Event-B. This particular model is the very first (abstract) model in the development chain.

The overall model is made of seven refinement steps with 470 verification conditions of which 301 were discharged automatically by Rodin theorem provers. Its development span over several months and several early versions were abandoned either due to misrepresentation of some railway concepts or unacceptable verification costs.

Apart from its role in the validation of first two layers, the discrete operational rules of the third layer are used to visually animate train movements over a given schema. There are two main applications for such an animation: replaying the results of model checking of discrete driving rules in order to pin-point the source of an error in a topology or a control table; and helping an engineer to understand how trains may travel through a schema with a given set of control rules.

4.3 Schema topology theory

The schema topology theory is responsible for verifying logical conditions expressed over track layout (i.e., track connections, point placement) and logical topology (i.e., routes and lines as paths through a schema). Few examples of verification conditions include the connectivity property (no isolated pieces of track), continuity of routes and lines, absence of cycles, correct traversal of points and valid placement of train detection circuit boundaries.

As a whole, these conditions express what we understand to be a valid track topology. We have tested them against a number of large-scale real-life layouts and we able to discover some problem in already informally validated track topologies. In addition, semi-automated alteration and generation of track layouts (e.g., via the improvement patterns we are developing in the tool) necessitates a careful and strict inspection of these basic properties. An automated verification process ensures high productivity and enables an engineer to explore a large range of designs within a short time.

```
machine route0
  sees ctx_line
  variables
    t_line   // Train/line association
    t_r_hd   // Train head position on a line
    t_r_tl   // Train tail position on a line
  invariant
    t_line ∈ TRAIN ⇸ LINE
      // A train is mapped to the id of a route occupied by the head of a train
    t_r_hd ∈ TRAIN ⇸ ℕ1
      // correspondingly, t_r_tl(t) is the id of the route occupied by the tail of train t
    t_r_tl ∈ TRAIN ⇸ ℕ1
    dom(t_line) = dom(t_r_hd)
    dom(t_line) = dom(t_r_tl)
      // A train occupies a continuous route interval of route from tail till head
    ∀t·t ∈ dom(t_line) ⇒ t_r_tl(t) .. t_r_hd(t) ≠ ∅
    The routes a train occupies are the routes defined by the train line
    ∀t·t ∈ dom(t_line) ⇒ t_r_tl(t) .. t_r_hd(t) ⊆ dom(Line(t_line(t)))
    // Initially, there are no trains in the system
  initialisation
    t_line, t_r_hd, t_r_tl := ∅, ∅, ∅
  events
      // A train may appear in the system with this operation
    appear =
    any t, l where
      t ∈ TRAIN \ dom(t_line)   // a train must be not already in the system
      l ∈ LINE
    then
      t_line(t) := lset the train line to l
      t_r_hd(t), t_r_tl(t) := 1, 1   // set head and tails routes
    end
      // Moves the head of a train from one route to another
    move_route_hd =
    any t where
      t ∈ dom(t_line)
      t_r_hd(t) < LineLen(t_line(t))   // train head must not be on the last line route
    then
      t_r_hd(t) := t_r_hd(t) + 1   // move the head one step forward
    end
      // Moves the tail of a train between routes
    move_route_tl =
    any t where
      t ∈ dom(t_line)
      t_r_tl(t) < t_r_hd(t)   // a tail must be strictly behind the head of the train
    then
      t_r_tl(t) := t_r_tl(t) + 1   // move the tail one step forward
    end
    ...
```

Fig.6. An Event-B model of abstract, route-level train movement (an excerpt)

Figure 7 gives a sample of verification conditions written in the Classical B notation (Abrial 1996) and ready to be processed by model checking tool ProB (Leuschel and Butler 2003). Not shown is the encoding of domain specific elements (track graph, control tables) as sets, relations and functions of a B model. For a real-life example, such a model may be 6-14 thousand lines long. The same conditions and constructs are also generated in the Why3 theory notation. It is not a direct translation of the B model and we intentionally use a different representation of relations and functions to introduce a form of modelling diversity. At the moment, for the topology theory, ProB and Why3 verifications chains deliver broadly similar performance.

```
/* (1)  */  {} <<: NODE  &
/* (2.a)  */  {} <<: TRACK  &
/* (2.b)  */  TRACK <: NODE * NODE  &
/* (2.c)  */  elm(TRACK) = NODE  & /* all nodes are connected by tracks */
...
/* (10) */ AMBIT : LA --> (POW(NODE) * POW(TRACK)) &
/* (11)  */ ! (a, q, p) . (a : ran(AMBIT) & a = ( q |-> p ) => p <: q * q & {} <<: p) &
/* (12) */ ! (a, q, p) . (a : ran(AMBIT) & a = ( q |-> p) => p~ <: p) &
/* (13) */ ! (a, q, p) . (a : ran(AMBIT) & a = ( q |-> p) =>
               ! (n) . (n : q => closure(p)[{n}] = q) ) &
/* (14)  */ union({p | # (a, q) . ( a : ran(AMBIT) & q <: NODE &
                p <: TRACK & a = ( q |-> p))}) = TRACK &
/* (15)  */ ! (a, b, r, s, t, q) . (a : ran(AMBIT) & b : ran(AMBIT) & a /= b &
               a = (r |-> s) & b = (t |-> q) => s /\ q = {}) &
...
```

Fig.7. Schema well-formedness rules (an excerpt)

4.4 Control table theory

When the topology is verified we can define the conditions of operational safety. These are derived, via a formal proof, from a set of discrete (inertia-less) train movement rules and expressed as a set of constraints over signalling rules.

In SafeCap, we depart from the convention of associating control rules with trackside signals. Instead, we consider a more general situation where different signalling rules are applied depending upon the ultimate train destination or train type and attach control logic to a pair of line and route. This permits, for instance, to model, on the same track, an express train using two-aspect signalling and a freight train travelling over the same routes but in a three or four aspect mode. Such an arrangement may be used to achieve an optimal balance between headway and average speed in a heterogeneous traffic mix. Given the fact that in UK trackside signals are going to be made obsolete by 2030 (TSLG 2012) this represents a fairly modest scenario of using virtual signals to improve capacity.

The control table theory demonstrates such properties as the absence of potential collision (as may happen, for instance, when a proceed aspect is given while a protected part of track is still occupied) and derailment (due to incorrect point setting or point movement under a train). Other properties relate to the danger or cir-

cular dependencies between signals, dependencies between multi-aspect signals and operation of auto-signals, conformance with an Automatic Train Protection system, and verification of point and signal based flank protection. Certain properties, notably approach speed control via the timed occupation of a track section, are not verified at this stage, as the formalisation at this layer does not capture train inertia. Speed limit conformance and other time-related properties are formulated at the final, most detailed layer.

A list of sample control table theory conditions is given in Figure 8. For the shown rules, the outer quantification selects a pair of a line and a route that define a list of control rules (one per aspect). This model includes the topology model[2]. Constraint solving is the primary verification strategy: we try to detect a contradiction between concrete data structures defining topology and control tables and the verification conditions. Again, the model is given in both B and Why3 notations although this time the Why3 verification route is not successful for larger examples. This is due to the weakness in our axiomatization of the B mathematical notation in Why3. We are currently working on building a library of Why3 lemmas to support translation from B to Why3 and this, we believe, should deliver a significantly better result. In addition, for any mid to large-scale schema we currently have to exclude the verification of flank protection properties, as these require complicated computations over track topology. We are working on a program that would output a proof term for each instance of flank protection property so that a theorem prover or a constraint solver would only have to check the elementary steps of a prepared proof.

```
...
/* @label (CT.1): A permissive signal may be lit only when all route ambits are clear */
! (l, r). (l |-> r : CTO_DOM => ! (n). (n : 1 .. RASPECT(l, r)-1 =>
                  routeambits(r) <: CT_CLEAR(l, r, n) )) &
/* @label (CT.2): A route with an overlap may have permissive signal only
 when its overlap is reserved and confirmed as clear  */
! (l, r). (l |-> r : ROVERLAP & r : dom(LINE(l)) => ! (n). (n : 1 .. RASPECT(l, r)-1 =>
                  TA[fst(ROUTE(LINE(l)(r)))] <: CT_CLEAR(l, r, n))) &
/* @label (CT.3.a): No point is set both normal and reverse */
! (l, r). (l |-> r : CTO_DOM => CT_NORMAL(l, r) /\ CT_REVERSE(l, r) = {} ) &
...
```

Fig. 8. Control table conditions (an excerpt)

5 Conclusions

The SafeCap offers an efficient tool for signalling engineers to design railway nodes (stations and junctions) while automatically checking the conformity of the signalling and topology against a range of validation criteria expressing operational safety and design integrity properties. Such level of automation enables rapid

[2] At the level, it is assumed that the topology theory has been verified and the topology constraints are turned into axioms

exploration of signalling designs in the pursuit of optimal capacity and performance stability.

The work on the SafeCap Toolset is now taken further in our new project SafeCap for FuTRO supported by Rail Safety and Standards Board. In this work we are developing a support for an integrated reasoning about capacity and energy of railway networks and nodes while ensuring whole systems safety.

Acknowledgments The work has been conducted as part of the UK EPSRC/Rail Safety and Standards Board SafeCap, EPSRC SafeCap-Impact and Rail Safety and Standards Board SafeCap for FuTRO projects. This work has been partially supported by the EPSRC/UK TrAmS-2 Platform Grant.

References

Abrial J-R (1996) The B-Book. Cambridge University Press

Abrial J-R (2006). Train systems. In Butler M J, Jones C B, Romanovsky A, Troubitsyna E (eds), Rigorous Development of Complex Fault-Tolerant Systems (FP6 IST-511599 RODIN project), LNCS 4157, Springer, 1-36

Abrial J-R (2010). Modelling in Event-B. Cambridge University Press

Abrial J-R, Mussat L (1998). Introducing Dynamic Constraints in B. In Proceedings of B'98: Recent Advances in the Development and Use of the B Method, LNCS 1393, Springer, 83-128

Bobot F, Filliatre J-C, Marche C, Paskevich A (2011). Why3: Shepherd your herd of provers. In Boogie 2011: First International Workshop on Intermediate Verification Languages, 53-64

Burdy L (1999). Automatic Refinement. In Proceedings of BUGM at FM'99

Essame D and Dolle D (2007). B in Large-Scale Projects: The Canarsie Line CBTC Experience. In Julliand J, Kouchnarenko O (eds.), B, LNCS 4355. Springer, 252-254

Fokkink W J, Hollingshead P R (1998). Verification of Interlockings: from Control Tables to Ladder Logic Diagrams. In Proceedings of the 3rd Workshop on Formal Methods for Industrial Critical Systems (FMICS'98)

Hagalisletto A M, Bjork J, Yu I C, Enger P (2007). Constructing and Refining Large-Scale Railway Models Represented by Petri Nets. IEEE Transactions on Systems, Man, and Cybernetics, Part C, 444-460

Iliasov A, Romanovsky A (2012). SafeCap domain language for reasoning about safety and capacity. Pacific-Rim Dependable Computing Conference (PRDC 2012). Niigata, Japan. IEEE CS

Janczura C W (1998). Modelling and Analysis of Railway Network Control Logic using Coloured Petri Nets. PhD thesis, School of Mathematics and Institute for Telecommunications Research, University of South Australia

Leuschel M, Butler M (2003). ProB: A Model Checker for B. In Keijiro A, Gnesi A, Dino M (eds.), Formal Methods Europe 2003, LNCS 2805. Springer, 855-874

OpenTrack simulator (2014). Website. Available at http://www.opentrack.ch/. Accessed 2014.

RailSys simulation platform (2014). Website. Available at http://http://www.rmcon.de. Accessed 2014

RGSOnline (2014). Railway Group Standards. Signalling Design: Control Tables. Rail Safety and Standards Board (RSSB). Available at http://www.rgsonline.co.uk/. Accessed 2014

Rodin (2014). Rigorous Open Development Environment for Complex Systems (RODIN). IST FP6 STREP project, online at http://rodin.cs.ncl.ac.uk/

Romanovsky A, Thomas M (2013). Industrial deployment of system engineering methods. Springer

TPTP (2014). Thousands of Problems for Theorem Provers. Available at www.tptp.org/. Accessed 2014

TSLG (2012). The Rail Technical Strategy (RTS). 2012. Available at http://www.futurerailway.org/RTS/Pages/Intro.aspx

Winter K (2002). Model Checking Railway Interlocking Systems. In Proceeding of the 25th Australian Computer Science Conference (ACSC 2002). Australian Computer Society, Inc. Darlinghurst, Australia, Australia

Winter K, Robinson N (2003). Modelling Large Railway Interlockings and Model Checking Small Ones. In Proceeding of the Australian Computer Science Conference (ACSC 2003). Australian Computer Society, Inc. Darlinghurst, Australia, Australia